Resumo

Rodrigo Lugão Ribeiro Pinto

Universidade Federal Fluminense

12 de Abril de 2018

Conteúdo

Verificação, Validação, Testes

- "Elaborar uma solução para o meu problema" não é a única tarefa que deve ser cumprida ao desenvolver um sistema.
- É comum cometer erros erros durante o desenvolvimento de um sistema. O que fazer para achar esses erros? (Debugging)
- Como garantir que o meu sistema está sem erros? Como garantir que ele resolve meu problema corretamente?
- Como garantir que o sistema atende aos requisitos necessários? Como saber se o sistema realmente faz o que foi proposto?

Verificação, Validação, Testes

- Testes:
  - Testar todas as condições do sistema pode ser impraticável.
  - Realizar testes pode não ser possível, dependendo do sistema.
  - Resultados de teste podem não ter informação suficiente sobre um problema do sistema (no caso de algum teste não passar) e nem garantem que o sistema está correto (no caso em que todos os testes passam).
- Verificação por prova:
  - Usa axiomas e regras de prova para provar a corretude do sistema.
  - Funciona para sistemas de estados finitos e infinitos.
  - Dependendo do sistema, conseguir uma prova pode ser difícil.
  - Para cada sistema, uma prova diferente.
  - Pouco automatizável.

Verificação, Validação, Testes

- Verificação de modelos (Model Checking):
  - O sistema é representado como um modelo. As propriedades são verificadas em cima desse modelo.
  - Explora exaustivamente todos o comportamentos possíveis.
  - Automático.
  - Em geral é mais fácil modelar um sistema e suas propriedades do que elaborar uma prova de que o sistema está correto/atende à alguma propriedade.
  - Funciona para sistemas de estados finitos e infinitos.
  - Tipicamente só serve para sistemas com estados finitos.

Model Checking

- O problema: dado um modelo e uma (ou mais) propriedades, verificar se o modelo satisfaz essas propriedades.
- A solução: percorrer todo o modelo, verificando se a propriedade vale. Se todo o modelo foi percorrido e a propriedade não foi violada, então aquele modelo satisfaz aquela propriedade.
- Como transformar o sistema em um modelo?
- Como representar as propriedades?

Model Checking

- O sistema pode ser visto da seguinte maneira:
  - Um conjunto com todos os estados possíveis;
  - Um conjunto com todas as transições entre estados que podem ocorrer;
  - Um outro conjunto que diz todas as propriedades de cada estado.
- Uma estrutura de Kripke!
- As propriedades a serem verificadas podem ser descritas como fórmulas lógicas.
- Podemos usar lógicas modais.
- Os Model Checkers a seguir foram feitos em Maude.
- Disponíveis em https://github.com/RodrigoLugao/MaudeStuff .
- Por que Maude?

Lógicas Dinâmicas

- Extensões da Lógica modal.
- Conceito de Ações ou Programas.
- Associa a cada Ação α os operadores $[\alpha]$ e $< \alpha >$.
- Então $[\alpha]$p significa "depois de qualquer execução de α, vale p" e $< \alpha >$. significa "depois de alguma execução de α, vale p".
- O modelo de uma lógica dinâmica geralmente é uma estrutura de Kripke comum, com as transições nomeadas (pelas Ações).
- Num sistema Multi-agentes, os agentes executam ações, então é interessante usar lógicas que tratam de ações.

PDL

- Possui as características descritas anteriormente.
- Alguns operadores sobre Ações: ";", "U", "*".
- "a;b" significa "uma execução de a seguida de uma execução de b".
- "aUb" significa "uma execução de a ou de b".
- "a*" significa "0 ou mais execuções de a".
- O modelo é o mesmo.

Descrição

- Faz uma busca em profundidade limitada.
- Dividido em três módulos:
  - O primeiro contém os Sorts que definem os mundos, ações, contém também regras de como os operadores sobre ações funcionam, etc.
  - O segundo contém algumas operações auxiliares, e operações que definem a relação de satisfação, ou seja, o "corpo" do Model Checker.
  - O terceiro módulo é a parte customizável, onde o usuário especifica o modelo a ser verificado.
- Depois de especificar seu modelo, o usuário verifica esse modelo com o comando "red modelCheck(W, F, N, empty) ."

Descrição

- O Model Checker cai em diferentes casos dependendo da fórmula fornecida.
  - Se não houver operador modal na fórmula, ele faz uma reflexão para o módulo do modelo e verifica se a proposição (ou fórmula) vale naquele mundo.
  - Se houver operador modal com uma única ação, o Model Checker faz uma reflexão para buscar os mundos vizinhos a partir daquela ação, e prossegue com a busca em profundidade em cada um dos vizinhos, um por vez. Se algum desses vizinhos retorna verdadeiro, esse passo também retorna verdadeiro (só verifica o "diamond").
  - Se houver operador modal com ações compostas, o Model Checker faz uma reflexão para o primeiro módulo, e busca todas as aplicações das regras sobre os operadores de ações possíveis. Uma busca em profundidade é feita em cada opção possível.

Descrição

- O Model Checker cai em diferentes casos dependendo da fórmula fornecida.
  - Se em algum momento a profundidade atinge o limite, então o Model Checker retorna verdadeiro.
  - Caso terminem os "ramos" da busca sem que um "ramo" verdadeiro tenha sido achado, então o Model Checker retorna verdadeiro.

DPL

- Lógica Deôntica de Ações (versão proposicional).
- Alguns operadores sobre Ações e fórmulas:
  - $\sqcap$ e $\sqcup$ são execução paralela e escolha não determinística, respectivamente.
  - $\bar{\alpha}$ é a execução de uma ação diferente de α e $\emptyset$ e U são a ação impossível e a execução de qualquer ação, respectivamente.
  - $\alpha =_{act} \beta$ indica que α e β fazem parte dos mesmos eventos.
  - $\mathsf{P}(\alpha)$ é a permissão forte, ela é verdadeira se a execução de α é permitida em qualquer cenário.
  - $\mathsf{P}_w(\alpha)$ é a permissão fraca, ela é verdadeira se a execução de α é permitida em algum cenário.
  - Os demais são os operadores clássicos.
  - Obs.: O operador de obrigação é definido da seguinte maneira: $\mathsf{O}(\alpha) \equiv \mathsf{P}(\alpha) \wedge \neg \mathsf{P}_w(\bar{\alpha})$.

DPL

- Conceito de Eventos: Uma Ação tem um conjunto de eventos dos quais ela faz parte.
- Algumas diferenças no Modelo:
  - As transições agora não são rotuladas por Ações, mas sim por Eventos.
  - O modelo tem uma função I que define o comportamento de alguns operadores sobre Ações e fórmulas, e define de que Eventos as Ações fazem parte.
  - O modelo também possui um conjunto P que define em que mundos os Eventos são permitidos.

Descrição

- Também faz uma busca em profundidade limitada.
- Também é dividido em três módulos, que funcionam de maneira parecida.
- Agora o usuário tem que especificar o conjunto P e o conjunto I do seu modelo.
- Depois de especificar seu modelo, o usuário verifica esse modelo com o comando "red modelCheck(parametros) ."
- A operação modelCheck não faz mais reflexão para o primeiro módulo, pois a função I já tem os casos com os diferentes operadores sobre ações.
- Mas faz algumas reflexões para o terceiro módulo a mais, para verificar as fórmulas P(a) e Pw(a).

Lógicas Dinâmicas (II)

- Têm em comum o conceito de ações, e os mesmos operadores modais sobre ações.
- Alguns elementos dos modelos (Conjunto de mundos ou estados, transições rotuladas, função de valoração, etc) também são compartilhados nas diferentes lógicas.
- Outros elementos do modelo podem variar.
- Operadores sobre Ações e fórmulas também variam.
- A função de satisfabilidade deve ter muita coisa em comum entre as diferentes lógicas.
- Fazer um Model Checker que possa ser modificado pelo usuário em função da lógica dinâmica que ele quer usar.

Descrição

- Também faz uma busca em profundidade limitada.
- Também é dividido em três módulos:
  - Os módulos precisam implementar algumas coisas pré definidas: o primeiro módulo precisa ter um conjunto de mundos, por exemplo.
  - O primeiro módulo agora deve ter todos os operadores sobre ações e/ou sobre fórmulas da lógica escolhida, bem como novos conjuntos e sorts que representam elementos da lógica, se necessário.
  - O segundo módulo não é alterado. Ele ainda é a parte da satisfabilidade, agora com casos mais genéricos.
  - Caso ele caia num operador que não conheça, ele faz uma reflexão para o primeiro módulo, para que o mesmo devolva uma fórmula ou conjunto de ações que ele conhece.
  - O terceiro módulo ainda tem o modelo. É obrigatório que ele tenha os elementos de modelo que são comuns entre as lógicas.

Descrição

- Também é dividido em três módulos:
  - A transição do modelo agora é rotulada por qualquer elemento que a lógica use como rótulo.
  - no terceiro módulo devem ficar os elementos adicionais do modelo da lógica escolhida.